Question 1

Please give short answers (1-3 sentences) to the following questions.

(a) What is the *curse of dimensionality*?

Very large feature spaces (i.e., high number of features) for which we might not have enough data to train our machine learning models.

(b) What is *overfitting* and how can we avoid it?

The risk of highly flexible (complicated) models
we can avoid through obtaining more data, reducing the number of features, or performing regularization.

(c) It is often assumed that the magnitude (i.e., $|w_1|$, $|w_2|$, ..., , $|w_D|$) of the linear regression coefficients $\mathbf{w} = [w_1, \ldots, w_D]$, where $y = \mathbf{w}^T\mathbf{x}$, indicates the importance of the corresponding features. Describe a situation where this may not be true.

The input features might not have the same range.

e.g., x_1 has range between $[0, 100]$
x_2 has range between $[0, 1]$

Question 2

Please select the correct answer(s) to the following questions. Multiple answers could be correct.

(a) Which are possible hyper-parameters?

[✓ A.] The learning rate α of gradient descent

[✓ B.] The weight of regularization λ in linear regression

[✓ C.] The degree of polynomial in non-linear regression with a polynomial function

D. The weights w in non-linear regression

Question 3

Assume a non-linear regression, whose output $y \in \mathbb{R}$ is predicted from the input $x \in \mathbb{R}$ as follows:

$$y = f(x) = bx^3 + c$$

The goal of the non-linear regression is to learn the weights $b, c \in \mathbb{R}$ from the training data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$.

Hint: For the following, you can use the formula: $(z_1+z_2+z_3)^2 = z_1^2+z_2^2+z_3^2+2z_1z_2+2z_2z_3+2z_1z_3$

(a) Write the residual sum of squares (RSS) error between the actual and predicted outcomes.

$$RSS(b,c) = \sum_{n=1}^{N} (y_n - f(x_n))^2 = \sum_{n=1}^{N} (y_n - bx_n^3 - c)^2$$

(b) Assuming that c is constant, derive the gradient descent formula for updating the weight $b \in \mathbb{R}$ so that the RRS error (as obtained from question a) is minimized with respect to b. Give the batch update formula and the stochastic update for a random sample (x_m, y_m). Set up your formulas so that the learning rate parameter is $\nu(k) > 0$, in which k is the iteration index.

$$\frac{\partial RSS(b,c)}{\partial b} = 2\sum_{n=1}^{N} (y_n - bx_n^3 - c)(-x_n^3) = -2\sum_{n=1}^{N} x_n^3 (y_n - bx_n^3 - c)$$

Batch: $b(k+1) := b(k) - \nu(k) \frac{\partial RSS(b(k), c)}{\partial b(k)}$

$$:= b(k) + \nu(k)\, 2 \sum_{n=1}^{N} x_n^3 (y_n - b(k)x_n^3 - c)$$

Stochastic: $b(k+1) := b(k) + \nu(k) \cdot 2 \cdot x_n^3 (y_n - b(k)x_n^3 - c)$ for random x_n

(c) Assuming that b is constant, derive the gradient descent formula for updating the weight $c \in \mathbb{R}$ so that the RRS error (as obtained from question a) is minimized with respect to c. Give the batch update formula and the mini-batch update for a random subsample of the training data $\mathcal{D}_s \in \mathcal{D}$. Set up your formulas so that the learning rate parameter is $\nu(k) > 0$, in which k is the iteration index.

$$\frac{\partial RSS(b,c)}{\partial c} = -2\sum_{n=1}^{N}(y_n - bx_n^3 - c)$$

Batch: $c(k+1) := c(k) + \nu(k)\, 2 \sum_{n=1}^{N} (y_n - bx_n^3 - c(k))$

Stochastic: $c(k+1) := c(k) + \nu(k)\, 2(y_n - bx_n^3 - c(k))$ for a random k.

(d) We now apply $l2$-norm regularization for the aforementioned non-linear regression in terms of both weights $b, c \in \mathbb{R}$. Write the expression of the evaluation criterion for the non-linear regression with regularization, including the RSS error and regularization term, assuming $\lambda > 0$ as the model complexity for both b and c.

$$RSS(b,c) = \sum_{n=1}^{N}(y_n - bx_n^3 - c)^2 + \lambda b^2 + \lambda c^2$$

(e) Derive the gradient descent formula for jointly updating weights $b, c \in \mathbb{R}$ so that the optimization criterion of the regularized non-linear regression (as obtained from question d) is minimized jointly with respect to b and c. Give the batch update formula using the learning

rate parameter $\nu(k) > 0$, in which k is the iteration index.
Hint: Assume a weight vector $\mathbf{w} = [b, c]^T \in \mathbb{R}^2$.

$$\frac{\partial RSS(b,c)}{\partial b} = -2 \sum_{n=1}^{N} x_n^3 (y_n - b x_n^3 - c) + 2\lambda b$$

$$\frac{\partial RSS(b,c)}{\partial c} = -2 \sum_{n=1}^{N} (y_n - b x_n^3 - c) + 2\lambda c$$

$$\begin{bmatrix} b(k+1) \\ c(k+1) \end{bmatrix} := \begin{bmatrix} b(k) \\ c(k) \end{bmatrix} - \nu(k) \begin{bmatrix} -2 \sum_{n=1}^{N} x_n^3 (y_n - b x_n^3 - c) + 2\lambda b \\ -2 \sum_{n=1}^{N} (y_n - b x_n^3 - c) + 2\lambda c \end{bmatrix}$$

Question 4

Let the random variable $\mathcal{X}$ follow an exponential distribution with parameter λ, i.e., the probability of x is $f(x) = \lambda e^{-\lambda x}$. Also let's assume a set of independent and identically distributed data (i.i.d.) samples $\mathcal{X} = \{x_1, \ldots, x_N\}$ which follows the exponential distribution

(a) Find the log-likelihood of the data.

$$\ell(\lambda) = \prod_{n=1}^{N} f(x_n) = \prod_{n=1}^{N} \lambda e^{-\lambda x_n}$$

$$\mathcal{L}(\lambda) = \log \prod_{n=1}^{N} f(x_n) = \sum_{n=1}^{N} (\log \lambda - \lambda x_n) = \sum_{n=1}^{N}$$

$$= N \log \lambda - \lambda \sum_{n=1}^{N} x_n$$

(b) Find the extreme point of the above likelihood with respect to λ.

$$\frac{\partial \mathcal{L}(\lambda)}{\partial \lambda} = \frac{N}{\lambda} - \sum_{n=1}^{N} x_n$$

$$\frac{\partial \mathcal{L}(\lambda)}{\partial \lambda} = 0 \Rightarrow \frac{N}{\lambda} - \sum_{n=1}^{N} x_n = 0 \Rightarrow \lambda = \frac{N}{\sum_{n=1}^{N} x_n}$$

Question 5

Assume that you have a set of training and test data, $\mathbf{X}^{\text{train}} \in \mathbb{R}^{D \times N_1}$ and $\mathbf{X}^{\text{test}} \in \mathbb{R}^{D \times N_2}$, respectively, with corresponding label vectors $\mathbf{y}^{\text{train}} \in \mathbb{R}^{N_1}$ and $\mathbf{y}^{\text{test}} \in \mathbb{R}^{N_2}$. You would like to perform classification using logistic regression and use cross-validation on the training set to determine the best value of the learning rate $\alpha = \{0.001, 0.01, 0.1\}$ for the gradient descent optimization of logistic regression. Please provide a basic pseudocode to do this. In the pseudocode, please also include the last evaluation step after having identified the best learning rate α based on the test data.

Hint: You can assume a function *pred*=LR(X,y,Z,α), which provides a decision for test samples Z using training data X, training labels y and gradient descent learning rate α, and a function *acc*=ComputeAcc(*pred*,*lab*), which computes the classification accuracy between predicted class *pred* and actual labels *lab*.

```
Split $\{X^{train}, y^{train}\}$ into S equal parts $\{X_s^{train}, y_s^{train}\}_{s=1}^{S}$
 A = [0.001, 0.01, 0.1]
 CrossValAcc = [ ]      (vector to store the accuracies from cross-validation)

 for   a in A
       E = [  ]  (vector to store accuracies within each
       for  s = {1, ..., S} :          set difference (all train data, except the ones from current fold s)
            pred = LR( $\underbrace{X^{train} \setminus X_s^{train}, y^{train} \setminus y_s^{train}}$ ,  $X_s^{train}$ , a)
            E(s) = Compute Acc(pred, $y_s^{train}$)
       CrossValAcc = [CrossValAcc  $\frac{1}{S}$[E(1)+...+E(S)]]

                              gives the index of the max element in vector CrossValAcc
 a_best = A($\overbrace{\text{argmax}(\text{CrossValAcc})}$)  (the best learning rate)

 pred = LR($X^{train}$, $y^{train}$, $X^{test}$, a_best)

 acc = Compute Acc (pred, $y^{test}$)
```